0. Ejemplo de configuración de red en SUSE con NetworkManager

Aquí defines cómo se conectará tu servidor físicamente a la red en openSUSE/SLES: enlaces redundantes (bonding), VLANs para segmentar el tráfico, y bridges para conectar máquinas virtuales y pods a distintas redes. Esta configuración es la base para un clúster flexible y segmentado.

> **Pre-requisitos:**
>
> - openSUSE/SLES con NetworkManager activo.
> - Interfaces físicas: `em1` y `em2` (ajusta según tu hardware).

---

1. Crear el bond (LACP 802.3ad, rápido, hash layer3+4)

```
nmcli con add type bond ifname bond0 mode 802.3ad
nmcli con mod bond0 bond.options
"mode=802.3ad,miimon=100,updelay=200,downdelay=200,lacp_rate=fast,xmit_hash_pol
icy=layer3+4"
```

2. Añadir interfaces físicas al bond

```
nmcli con add type ethernet ifname em1 master bond0
nmcli con add type ethernet ifname em2 master bond0
```

3. Crear VLANs sobre el bond

```
nmcli con add type vlan ifname vlan10 dev bond0 id 10
nmcli con add type vlan ifname vlan20 dev bond0 id 20
nmcli con add type vlan ifname vlan30 dev bond0 id 30 ip4 192.168.3.2/24
nmcli con add type vlan ifname vlan40 dev bond0 id 40 ip4 192.168.4.2/24
```

4. Crear bridges y unirlos a las VLANs

```
# Bridge de administración (br-admin) sobre vlan10
nmcli con add type bridge ifname br-admin
nmcli con add type bridge-slave ifname vlan10 master br-admin
nmcli con mod br-admin ipv4.addresses 192.168.0.42/24
nmcli con mod br-admin ipv4.method manual
nmcli con mod br-admin ipv4.gateway 192.168.0.1
nmcli con mod br-admin ipv4.dns "192.168.0.1 1.1.1.1 8.8.8.8"
```

```
# Bridge de servicios (br-srv) sobre vlan20
nmcli con add type bridge ifname br-srv
nmcli con add type bridge-slave ifname vlan20 master br-srv
nmcli con mod br-srv ipv4.addresses 192.168.200.2/22
nmcli con mod br-srv ipv4.method manual
```

5. Sube todas las conexiones (en orden: bond, VLANs, bridges)

```
nmcli con up bond0
nmcli con up vlan10
nmcli con up vlan20
nmcli con up vlan30
nmcli con up vlan40
nmcli con up br-admin
nmcli con up br-srv
```

Guía rápida: Instalar Kubernetes en openSUSE/SLES

Esta guía cubre todos los pasos necesarios para instalar Kubernetes en openSUSE Leap, Tumbleweed o SLES, usando containerd como runtime y gestionando todo con zypper. Sigue el orden de los pasos para evitar problemas.

Guía definitiva de despliegue Kubernetes (openSUSE/SLES)

> **Revisión y correcciones basadas en el history real de despliegue**

1. Prerrequisitos del sistema (SUSE)

Este paso parte de una instalación limpia de openSUSE/SLES actualizada y con permisos de administrador. Instala utilidades básicas, configura el repositorio oficial de Kubernetes y desactiva SWAP, igual que harías en Ubuntu, pero con comandos adaptados a zypper y la gestión de repositorios en SUSE.

a) Actualiza el sistema y paquetes básicos

```
sudo zypper refresh
sudo zypper update
sudo zypper install -y curl ca-certificates keepalived chrony
```

b) Añade el repositorio oficial de Kubernetes

Crea el archivo de repositorio para Kubernetes (v1.33):

```
cat <<EOF | sudo tee /etc/zypp/repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.33/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.33/rpm/repodata/repomd.xml.key
EOF
```

Actualiza la caché de repositorios y acepta la clave GPG cuando se te pida:

```
sudo zypper refresh
sudo zypper update
```

> Cuando veas el aviso sobre la clave GPG, pulsa 'a' para aceptar siempre.

c) Configuración de Keepalived

En SRVFKVM01 (MASTER):

```
cat <<EOF | sudo tee /etc/keepalived/keepalived.conf
vrrp_instance VI_1 {
    state MASTER
    interface br-admin
    virtual_router_id 51
    priority 150
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 42manabo42
    }
    virtual_ipaddress {
        192.168.0.20/24
    }
}
EOF
```

En los demás (BACKUP):

```
cat <<EOF | sudo tee /etc/keepalived/keepalived.conf
    vrrp_instance VI_1 {
        state BACKUP
        interface br-admin
```

```
        virtual_router_id 51
        priority 100
        advert_int 1
        authentication {
            auth_type PASS
            auth_pass 42manabo42
        }
        virtual_ipaddress {
            192.168.0.20/24
        }
    }
EOF
```

Después en todos:

```
sudo systemctl enable keepalived
sudo systemctl start keepalived
```

d) Sincronización horaria

En todos los servidores del cluster:

```
sudo systemctl enable chronyd
sudo systemctl start chronyd
sudo chronyc makestep  # Fuerza la sincronización inicial
```

---

2. Instala containerd (runtime recomendado)

`containerd` es el motor que gestiona los contenedores en el clúster. Kubernetes necesita un "runtime" para crear y controlar los pods, y containerd es la opción oficial y más estable.

```
sudo zypper install -y containerd
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/'
/etc/containerd/config.toml
sudo systemctl restart containerd
sudo systemctl enable containerd
```

---

3. Desactiva SWAP (requisito para Kubernetes)

Kubernetes requiere que el intercambio de memoria (swap) esté desactivado. Esto evita problemas de rendimiento y estabilidad.

```
sudo swapoff -a
sudo sed -i '/ swap / s/^/#/' /etc/fstab
```

---

4. Prepara el kernel y sysctl y abre puertos

En este paso se habilitan módulos y parámetros de red necesarios para que Kubernetes gestione correctamente el tráfico entre pods y nodos.

```
sudo modprobe overlay
sudo modprobe br_netfilter
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
sudo sysctl --system
```

el trafico basico que se debe permitir es:

```
# Básico Kubernetes
# Asocia la interfaz de red interna del clúster a la zona trusted (típicamente
br-admin)
sudo firewall-cmd --zone=trusted --add-interface=br-admin --permanent
sudo firewall-cmd --zone=trusted --add-interface=vlan40 --permanent

# Abre solo en zona trusted los puertos necesarios para el clúster:
sudo firewall-cmd --zone=trusted --add-port=6443/tcp --permanent    # API
Server
sudo firewall-cmd --zone=trusted --add-port=10250/tcp --permanent   # Kubelet
API
sudo firewall-cmd --zone=trusted --add-port=2379/tcp --permanent    # etcd
client (solo control-plane)
sudo firewall-cmd --zone=trusted --add-port=2380/tcp --permanent    # etcd peer
(solo control-plane)
sudo firewall-cmd --zone=trusted --add-port=8472/udp --permanent    # Flannel
VXLAN (si usas Flannel)

# Si vas a exponer servicios (Traefik, MetalLB), abre HTTP/HTTPS solo si lo
necesitas en todos los nodos
# (mejor hacerlo en los nodos donde se expone tráfico externo, no en todos)
sudo firewall-cmd --zone=trusted --add-port=80/tcp --permanent
sudo firewall-cmd --zone=trusted --add-port=443/tcp --permanent
```

```
# (Opcional) Si usas NodePort, abre el rango solo si usas ese tipo de servicio:
sudo firewall-cmd --zone=trusted --add-port=30000-32767/tcp --permanent

# Recarga las reglas
sudo firewall-cmd --reload
```

---

5. Instala los componentes de Kubernetes (kubectl, kubelet, kubeadm)

Instala en un solo paso las utilidades principales:

```
sudo zypper install -y kubectl kubelet kubeadm
```

Edita `/etc/sysconfig/kubelet` para poner la dirección IP que tendrá el nodo:

```
KUBELET_EXTRA_ARGS=--node-ip=192.168.4.1   # <-- Dirección ip de cada nodo en
la Vlan interna del cluster
```

Activa kubelet para que se inicie automáticamente:

```
sudo systemctl enable kubelet
sudo systemctl start kubelet
```

> Es **normal** que el servicio kubelet falle en bucle hasta que inicialices el clúster con `kubeadm init`.
> El error más frecuente es:
>
> - "failed to load Kubelet config file, path: /var/lib/kubelet/config.yaml"
>
> Tras ejecutar `kubeadm init`, kubelet arrancará correctamente.

5b. (Opcional) Habilita cgroup v2 (solo SLES si necesario)

Si necesitas cgroup v2 en SLES, añade esto al arranque del kernel (edita `/etc/default/grub`):

```
GRUB_CMDLINE_LINUX_DEFAULT="systemd.unified_cgroup_hierarchy=1 ..."
```

Y regenera el grub:

```
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

Reinicia para aplicar los cambios.

---

6. Inicializa el clúster (openSUSE/SLES)

Este paso crea el clúster de Kubernetes en el nodo principal (control-plane) sobre openSUSE/SLES. Aquí defines la red interna para los pods y la interfaz/VLAN física para el tráfico overlay del clúster, según tu diseño.

---

a) Inicializa el clúster especificando red de pods, la IP interna y la VIP

> **Importante:**
>
> - Usa la opción `--apiserver-advertise-address` para forzar que el nodo control-plane escuche en la IP de la VLAN interna de clúster (ejemplo: `192.168.4.x` en tu VLAN 40).
> - Usa `--control-plane-endpoint` para indicar la IP virtual compartida (VIP) gestionada por Keepalived. Debe ser una IP accesible por todos los nodos control-plane (ejemplo: `192.168.0.20`).
> - Usa `--apiserver-cert-extra-sans` para añadir la VIP (y/o el FQDN) como SAN en el certificado TLS del API server, así todos los nodos confían en esa IP/nombre.
> - Usa `--pod-network-cidr=10.244.0.0/16` si vas a usar Flannel como CNI (ajusta si usas otro CNI).

```
sudo kubeadm init \
  --apiserver-advertise-address=192.168.4.1 \
  --control-plane-endpoint="192.168.0.20:6443" \
  --apiserver-cert-extra-sans="192.168.0.20" \
  --pod-network-cidr=10.244.0.0/16 \
  --upload-certs
```

Cambia `192.168.4.1` por la IP interna del nodo donde ejecutas el comando (en la VLAN de clúster) y `192.168.0.20` por la IP VIP gestionada por Keepalived.

---

c) Configura kubectl para tu usuario

Permite usar el comando `kubectl` como usuario normal copiando la configuración de administración del clúster a tu carpeta personal.

En el nodo master (SRVFKVM01)

```
mkdir -p $HOME/.kube
cd /etc/kubernetes
sudo cp -i admin.conf $HOME/.kube/config
sudo scp config admin.c3s@192.168.0.4X:/home/admin.c3s/admin.conf    #a todos
```

```
los nodos
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

d) Instala la red de pods (Flannel) usando la VLAN interna del clúster

Kubernetes solo define la infraestructura; necesitas un complemento de red (CNI) para que los pods puedan comunicarse entre sí. Flannel es la opción más sencilla y compatible, y puedes configurarla para usar una interfaz/VLAN específica para el tráfico overlay (muy recomendable si segmentas redes en tu clúster).

¡ATENCIÓN! CNI Y PLUGINS:

Antes de aplicar Flannel, **asegúrate de tener los plugins CNI instalados en** `/opt/cni/bin/` (en SUSE esto NO siempre lo hace el paquete de Flannel):

```
CNI_VERSION="v1.4.1"
ARCH="amd64"
curl -Lo cni-plugins.tgz
https://github.com/containernetworking/plugins/releases/download/${CNI_VERSION}
/cni-plugins-linux-${ARCH}-${CNI_VERSION}.tgz
sudo mkdir -p /opt/cni/bin
sudo tar -C /opt/cni/bin -xzf cni-plugins.tgz
```

Verifica que existen al menos:

- `/opt/cni/bin/loopback`
- `/opt/cni/bin/flannel`
- `/opt/cni/bin/bridge` y otros

Corrige permisos si hace falta:

```
sudo chmod +x /opt/cni/bin/*
sudo chown root:root /opt/cni/bin/*
```

Luego sí aplica Flannel:

```
kubectl apply -f https://github.com/flannel-
io/flannel/releases/latest/download/kube-flannel.yml
```

---

e) Comprueba que Flannel funciona

```
kubectl -n kube-flannel get pods -o wide
```

> Todos los pods deben estar en estado `Running`.

Si no levanta, **verifica logs y revisa `/opt/cni/bin/`**:

- Los binarios deben ser para la arquitectura correcta (x86_64/amd64).
- Deben tener permisos de ejecución y ser propiedad de root.

---

f) Une el resto de nodos control-plane

En los demás nodos control-plane (tras configurar Keepalived y tener la VIP activa):

```
sudo kubeadm join 192.168.0.20:6443 \
    --token <TOKEN> \
    --discovery-token-ca-cert-hash <HASH> \
    --control-plane \
    --certificate-key <CERTIFICATE_KEY> \
    --apiserver-advertise-address=192.168.4.2
```

> **Importante:** Cambia la IP `192.168.4.2` por la correspondiente a cada servidor control-plane que estés uniendo.

Recuerda usar los valores correctos de token, hash y certificate-key que te dará el comando `kubeadm init` o puedes volver a consultar desde el nodo principal:

```
kubeadm token create --print-join-command --control-plane
```

Credenciales para todos (lo que le gusta al jefe):

```
mkdir -p $HOME/.kube
cd
sudo cp -i admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

EXTRA: Si te ha caducado el TOKEN, o no lo guardaste

Cuando el token de unión (`kubeadm join`) ha caducado, sigue estos pasos en el **nodo principal (control-plane)** para generar un nuevo token y el comando actualizado:

```
# 1. Sube los certificados y obtén la nueva clave de certificado (certificate-
key)
sudo kubeadm init phase upload-certs --upload-certs
```

La salida mostrará la nueva `certificate key`, necesaria para añadir un nuevo nodo de control-plane.

```
# 2. Genera un nuevo token y muestra el comando kubeadm join listo para usar
sudo kubeadm token create --print-join-command
```

La salida mostrará el comando `kubeadm join` con el nuevo token y el hash actual del CA.

Ejemplo de salidas:

```
[upload-certs] Using certificate key:
588ee9d0441c0aea36b2a2a638beae4cfa81dd3498eb61205c8496f344f6c55d

kubeadm join 192.168.0.20:6443 --token s9nlsp.gasv66tkc43zpiok --discovery-
token-ca-cert-hash
sha256:d41f0edb90c66c0555bdf4feca55f1e69019764be0fcd649a254e99ff124568f
```

Comando completo para unir un nodo **como control-plane**

Añade los parámetros `--control-plane --certificate-key <certificate-key> --apiserver-advertise-address=192.168.4.2` al final del comando `kubeadm join` generado. Usando el ejemplo anterior, el comando final sería:

```
sudo kubeadm join 192.168.0.20:6443 \
  --token s9nlsp.gasv66tkc43zpiok \
  --discovery-token-ca-cert-hash
sha256:d41f0edb90c66c0555bdf4feca55f1e69019764be0fcd649a254e99ff124568f \
  --control-plane \
  --certificate-key
588ee9d0441c0aea36b2a2a638beae4cfa81dd3498eb61205c8496f344f6c55d \
  --apiserver-advertise-address=192.168.4.2
```

Recuerda: Cambia `192.168.4.2` por la IP real del nodo que se está uniendo.

Con esto podrás unir cualquier nodo extra como nuevo control-plane, incluso cuando el token original haya caducado.

Troubleshooting y buenas prácticas extra

- Si ves errores con el loopback CNI (`exec format error`, `permission denied`, etc.), **borra el contenido de `/opt/cni/bin/` y vuelve a instalar los plugins**.
- Siempre revisa que el módulo `br_netfilter` está cargado y los sysctl activos (`lsmod | grep br_netfilter`).
- Si el nodo aparece como NotReady, revisa `/etc/cni/net.d/`, los logs de kubelet, y repite la reinstalación del CNI.
- **Si limpias todo y reinicias, repite estos pasos:**
    - Reinstala CNI plugins
    - Aplica Flannel
    - Reinicia kubelet